KARELIA-AMMATTIKORKEAKOULU

Tietojenkäsittelyn koulutusohjelma

Lasse Hurri

VERKKOMONINPELN TOTEUTTAMINEN PUN-TYÖKALULLA UNITY3D-PELIMOOTTORILLA TOTEUTETTUUN PELIIN

Opinnäytetyö

Toukokuu 2018

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ

Toukokuu 2018

Tietojenkäsittelyn koulutusohjelma

Tikkarinne 9

80200 JOENSUU

+358 13 260 600 (vaihde)

Tekijä

Lasse Hurri

Nimeke

Verkkomoninpelin toteuttaminen PUN-työkalulla Unity3D-pelimoottorilla toteutettuun peliin

Toimeksiantaja

Firetail Games Oy

Tiivistelmä

Tässä opinnäytetyössä aloitettiin toteuttamaan verkkomoninpeliä Firetail Games Oy:n kehittämään peliin nimeltä Drunkenpants. Verkkomoninpelin suunnittelu ja kehitys toteutettiin Firetail Gamesin henkilökunnan kanssa, mutta päävastuu oli opinnäytetyön tekijällä.

Verkkomoninpelin toteutustavasta ja työkalusta päätti kuitenkin toimeksiantaja. Peli on toteutettu Unity3D-pelimoottorilla. Verkkomoninpelin kehitykseen valittiin Exit Gamesin luoma ja ylläpitämä Photon Unity Networking-työkalu, joka soveltuu hyvin Unity3D-pelimoottorille.

Verkkomoninpelin vaaditut toiminnallisuudet sisälsivät helposti lähestyttävän käyttöliittymän sekä pelin toimivuuden verkon ylitse. Käyttöliittymän tuli olla mahdollisimman yksinkertainen ja käyttäjäystävällinen, jotta pelaajat pääsisivät mahdollisimman nopeasti peliin.

Opinnäytetyön tavoitteena oli luoda ainakin osittain toimiva verkkomoninpeli, jota voitaisiin helposti kehittää muutoksien ilmetessä. Tavoitteena oli myös kehittää omaa osaamista ja tutustua täysin uuteen teknologiaan ja työkaluun.

Opinnäytetyön aikana toteutettiin pelille uutta koodipohjaa, mikä vei paljon aikaa verkkomoninpelin toteutukselta. Tämän vuoksi useat toiminnallisuudet verkkomoninpeliin jäivät vielä toteuttamatta. Tästä huolimatta toteutuksessa päästiin tavoitteisiin ja pelaajat voivat luoda pelin verkon yli ja pelata keskenään.

Kieli

suomi

Sivuja 42

Asiasanat

verkkomoninpeli, moninpeli, pelaaminen, Unity, Photon, Exit Games, peliohjelmointi, pelisuunnittelu, käyttöliittymä.

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS

May 2018

Bachelor of Business IT

Tikkarinne 9

80200 JOENSUU

Author

Lasse Hurri

Title

Implementing an Online Multiplayer Feature Using PUN Tool to a Unity3D Game

Commissioned by

Firetail Games Oy

Abstract

The purpose of this thesis was to plan and start creating the online multiplayer feature to an existing game called Drunkenpants. The project was commissioned by Firetail Games Oy. The online multiplayer was designed and developed with the Firetail Games staff. The main responsibility was still on the thesis writer.

The commissioner decided the developer tools for the online multiplayer. The game is developed with Unity3D game engine and the online multiplayer is developed with Exit Game's Photon Unity Networking tool which can be easily applied to the Unity3D game engine.

The features of the online multiplayer will include easily approachable user interface and the basic game features over the internet. The UI should be simple and user-friendly so players would be able to start a game as fast as possible.

The main goal to achieve in this thesis were to create at least a partially working online multiplayer which would be easy to develop further. Another goal was to develop own skills and knowledge with learning a new technology and tool.

A new code base for the game was developed while the thesis was written. This caused delay in implementing online features and many of them weren't implemented by the time the thesis was finished. Despite this the goals were achieved; players can start a game and players can play against each other over the internet.

Language

Finnish

Pages 42

Keywords

online multiplayer, multiplayer, gaming, Unity, Photon, Exit Games, game programming, game design, user interface

Sisältö

1 Johdanto 5

2 Moninpelaaminen verkossa 5

2.1 Verkkopohjainen moninpelaaminen 6

2.1.1 Viive ja optimointi 8

3 Photon-pilvipalvelu 9

3.1 Photon palvelun rakenne 10

3.2 Palvelutarjonta 11

3.3 Photon Quantum ja deterministinen prosessointi 12

3.4 Photon Bolt 13

3.5 Photon PUN 14

3.6 Verkkoalustan valinta 17

4 Moninpelin suunnittelu 19

4.1 Moninpelin vaiheistuksen suunnittelu 20

4.2 Hahmon ja nimen asettaminen 20

4.3 Aula 22

4.4 Huoneet 22

5 Toteutus 23

5.1 PUN asentaminen ja konfigurointi 23

5.2 Yhdistäminen palvelimeen ja perus asetusten säätäminen 26

5.3 Toiminnollisuuksien toteutus 27

5.3.1 Graafinen käyttöliittymä 27

5.3.2 Pelaajan nimen asettaminen ja hahmovalinta 28

5.3.3 Huoneiden luonti, listaus ja peliin liittyminen 30

5.3.4 Huoneen sisältö 32

5.4 Pelin toiminnallisuudet ja tekniikat 33

5.4.1 Remote Procedure Calls 33

5.4.2 Jatkuva tiedonlähetys 34

5.4.3 RaiseEvent-tapahtumakutsut 37

6 Tulokset 38

7 Pohdinta 39
Lähteet 41

1 Johdanto

Tämän toiminnallisen opinnäytetyön tarkoituksena oli aloittaa verkkomoninpelin kehittäminen Firetail Games Oy -peliyritykselle kehitysvaiheessa olevaan Drunkenpants-peliin. Olin pelin kehityksen alkuvaiheissa mukana, mutta siirryin pois tehtävistä noin vuosi sitten. Opinnäytetyötä kirjoittaessa peli oli kehittynyt runsaasti ja se tarvitsi pelaajalle helposti lähestyttävän verkkomoninpelin.

Drunkenpants-peli toteutetaan Unity-pelimoottorilla ja yritys valitsi verkkomoninpelin kehittämiseen työkaluksi Photon Enginen tuotteen Photon Unity Networking (PUN). PUN-työkalulla kehitettyjä verkkomoninpelejä on useita ja monet kehittäjät kehuvat sen soveltuvan niin aloitteleville yrityksille kuin suuremmillekin sen helppokäyttöisyyden, hinnoittelun ja laajennettavuuden vuoksi. Exit Games, Photon Enginen luoja, tarjoaa kattavan dokumentoinnin, kuinka PUN-työkalun avulla voidaan luoda verkkomoninpeli. PUN on myös alustariippumaton, mikä tarkoittaa, että peliä on mahdollista pelata eri pelikonsoleilla tai pelilaitteilla keskenään verkon ylitse yhtäaikaisesti.

Opinnäytetyön tavoitteena oli luoda ainakin osittain toimiva verkkomoninpeli, jota voitaisiin helposti kehittää muutoksien ilmetessä. Tavoitteena oli myös kehittää omaa osaamista ja tutustua täysin uuteen teknologiaan ja työkaluun.

2 Moninpelaaminen verkossa

Moninpelaaminen tarkoittaa sitä, kun kaksi tai useampi pelaajaa pelaavat samaa peliä yhdenaikaisesti samassa peliympäristössä. Moninpelaamisen muotoja on pelaamisen historiassa nähty useanlaisia. Ennen internetin aikakautta moninpelaamiseen käytettiin useimmiten jaetun näytön moninpelaamista. Ensimmäisiin moninpeleihin kuului mm. vuonna 1958 tehty “Tennis for Two”. Kyseessä oli peli, jota pelattiin analogisella tietokoneella. Näyttönä käytettiin oskilloskooppia ja ohjaimina kahta kustomoitua alumiinista ohjainta. Varhaisen pelihistorian ehkä tunnetuimpia moninpelejä on Pong, jonka Atari julkaisi vuonna 1972. (Wikipedia 2018.)

2.1 Verkkopohjainen moninpelaaminen

Verkkopelaamisella tarkoitetaan kahden tai useamman pelaajan pelaamista keskenään, joko paikallisessa LAN-verkossa tai internetin välityksellä. Verkkopelaaminen mahdollistaa pelaajien pelaamisen omilla laitteillaan eikä vaadi fyysistä läsnäoloa.

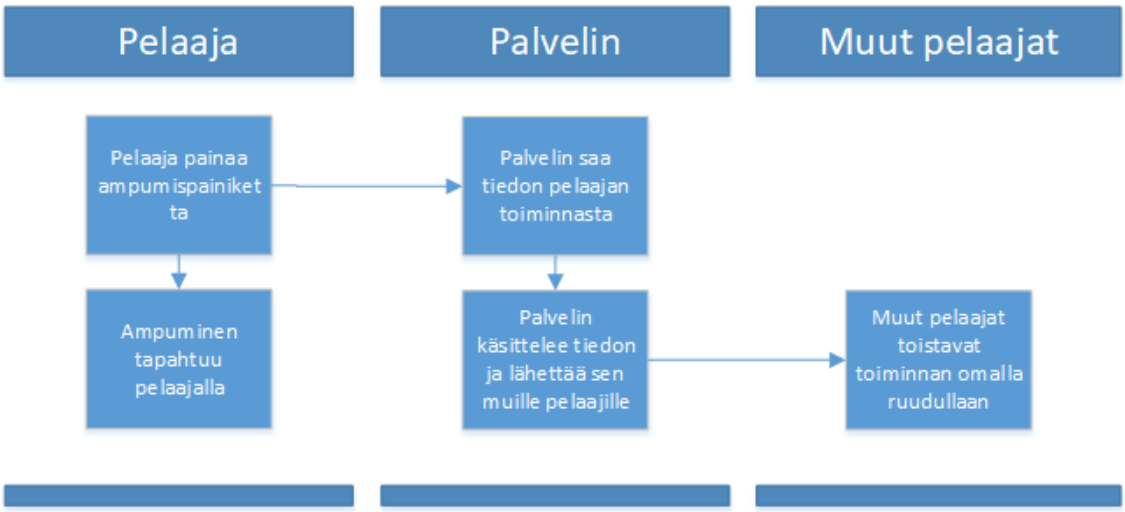
Internetissä verkkopelaaminen vaatii käyttäjältä yhteyden internetiin ja yhteyden palvelimeen, jonne pelaaja lähettää viestejä ja palvelin välittää ne muille samaan palvelimeen yhdistyneille pelaajille. Yhteys internetin välityksellä luodaan käyttämällä verkon yhteyspisteitä, jotka toimivat palvelinpuolen tiedon välittäjinä.

Verkkopelaamisessa voidaan käyttää useita eri tekniikoita ja kehitysalustoja. Verkkopelaaminen vaatii aina jonkinlaisen palvelimen tai yhteyspisteen pelaajien välille, jotta pelaaja voiva kommunikoida keskenään. Palvelin voi olla joko erillinen tai yhteys voi olla suoraan pelaajien välillä. Tätä kutsutaan P2P eli peer-to-peer -yhteydeksi. Tällöin tiedon välittäjänä toimivat pelaajat eikä palvelinta välttämättä tarvita heidän välille. (Bevilacqua 2013.)

On myös mahdollista, että pelaaja ylläpitää peliä, eli toimii ns. palvelimena pelaamisen ohella. Puhutaan ns. ”hostaamisesta”. Tällaisessa tilanteessa pelaaja luo pelin ja toimii kaikkien pelaajien välillä tiedonvälittäjänä. Kyseinen tapa luoda verkkopeli on monesti epävakaa ja vaatii käyttäjän tietokoneelta nopeaa prosessointitehoa sekä vakaata ja nopeaa verkkoyhteyttä. Mikäli kuitenkin ylläpidettävä verkkopeli ei vaadi paljoa ylläpitäjän koneelta ja verkkoyhteydeltä, hostaaminen voi olla jopa edullisin ja paras/helpoin ratkaisu kehittäjän kannalta. Tällöin myös pelin toimiminen verkossa ei ole riippuvainen palveluntarjoajasta.

Kuitenkin kun tapahtumat muuttuvat pelissä reaaliaikaisesti, verkkopelin ylläpitäjä saa edun muihin pelaajiin nähden, koska muutokset tapahtuvat hänen koneellaan viiveettä. Tämä johtuu siitä, että tieto liikkuu aina ylläpitäjän kautta ja hän saa ensimmäisenä tiedon tapahtuneista muutoksista. Vuoropohjaisissa peleissä pelin ylläpitäjä ei saa etua muihin pelaajiin nähden.

Erillinen palvelin tarkoittaa yritysten ylläpitämiä palvelimia palvelinsaleissa. Kyseiset palvelimet ovat tarkoitettuja toimimaan pelin ulkopuolella ja niiden tarkoitus on vain välittää tietoa. Palvelin ei käsittele peliin sisältyviä graafisia muutoksia, vaan ne välittävät puhdasta tietoa mitä muut pelaajat tekevät. Kaikki verkon ylitse tuodut viestit aiheuttavat muutoksia pelaajien pelaamassa pelissä ja muuttavat pelin kulkua sekä graafista ilmettä tietoa välittämällä. Esimerkiksi jos pelaaja painaa ampumapainiketta, lähetetään tieto siitä palvelimelle. Palvelin käsittelee saadun viestin ja välittää muille pelaajille, että kyseinen pelaaja ampuu ja se toistetaan samanaikaisesti muiden pelaajien ruudulla (kuva 1).



Kuva 1. Tiedonvälitys pelaajan ampumisesta.

Pelinkehityksen puolesta palvelimen puolella tarvitaan oma ohjelmisto, mikä käsittelee kaikkia kutsuja, joita pelaajat lähettävät palvelimelle. Pelin täytyy myös osata luoda yhteys palvelimeen ja lähettää haluttuja kutsuja palvelimelle. Palvelin välittää tiedon takaisin tiedon lähettäjälle ja muille kanssapelaajille. Pelistä riippuen palvelinyhteyksien tarpeellinen nopeus vaihtelee. Mikäli pelin on tarkoitus siirtää paljon dataa (viestejä) pelaajien välillä, tulee yhteyksien olla mahdollisimman nopeita ja datapakettien hyvin pakattuja. Ainoastaan internetyhteyden nopeus ei kuitenkaan riitä, vaan myös palvelimen tulee olla prosessointinopeudeltaan tehokas. Mikäli pelissä on yhtäaikaisesti 100 pelaajaa ja jokainen lähettää satoja viestejä samaan aikaan, vaatii se palvelimelta todella paljon tehoja.

2.1.1 Viive ja optimointi

Useimmissa verkkopeleissä on tärkeää, että palvelimen ja pelaajan välinen yhteys on nopea ja palvelimen sijainti ei ole kaukana pelaajasta. Mikäli yhteydessä ilmenee hitauksia, puhutaan viiveestä eli lagista, joka vaikeuttaa pelaajan reagointia ajoissa nopeisiin tilanteisiin. Viive aiheuttaa kanssapelaajille haittaa, sillä heille tieto pelaajan sijainnista päivittyy myöhään ja on mahdotonta ennakoida ja tietää, missä kohdassa pelaaja kyseisellä hetkellä sijaitsee. (Xicota 2018.)

Exit Games tarjoaa muiden palveluntarjoajien ohella useita palvelimia useissa eri maissa, joten yhteydet pitäisi pelaajien välillä olla suhteellisen hyvät, mikäli välimatkat eivät ole liian kaukaisia. Esimerkiksi Euroopan sisällä pelaavilla pelaajilla ei pitäisi olla minkäänlaisia ongelmia viiveen kanssa läheisen sijainnin vuoksi. Kuitenkin esimerkiksi tilanteessa, jossa peli on ylläpidetty Aasiassa ja pelaaja liittyy Euroopasta, voi aiheuttaa suuriakin hitauksia yhteydessä. (Exit Games 2018h.)

Kehitysvaiheessa pyrittiin ajattelemaan jatkokehityksen sekä optimoinnin haasteita. Tarkoituksena on luoda mahdollisimman saumaton ja nopeasti toimiva pelikokemus verkon yli pelaajille.

Optimoinnissa tulee ottaa huomioon, mitkä asiat ovat tarvittavia siirtää verkon ylitse ja kuinka esimerkiksi graafiset muutokset toteutetaan jokaiselle pelaajalle. Optimaalisin tilanne muutoksien toteuttamiseen on lähettää vain käsky pelaajan pelille ”suorita muutos X” ja pelaajan kone suorittaa muutoksen. Asia jota tulisi välttää mahdollisimman paljon, on lukuisien peliobjektien luominen (instantointi) samaan aikaan. Instantiointi voi olla todella raskas prosessi, jos sitä tapahtuu useita kertoja sekunnin sisällä. Raskaus riippuu paljon objektin suuruudesta ja monimutkaisuudesta. Suurien määrien luominen peliin on syytä tehdä siinä vaiheessa, kun peli alkaa. Pelissä tarvittavat peliobjektit luodaan ja varastoidaan etukäteen ja kun haluttu peliobjekti halutaan tuoda esille, se aktivoidaan näkyväksi.

3 Photon-pilvipalvelu

Photon on saksalaisen Exit Gamesin julkaisema alustariippumaton kehitystyökalu/pilvipalvelu, joka mahdollistaa verkkomoninpelien kehittämisen useille eri alustoille. Alustariippumaton ratkaisu mahdollistaa useiden eri laitteiden välisen kommunikoinnin ja pelaamisen keskenään. Photon tukee tällä hetkellä kymmentä eri alustaa, joihin kuuluvat mm. Android, Windows sekä Applen käyttöjärjestelmät iOS ja MAC OSx. (Exit Games 2018a.)

Alustariippumattomuuden lisäksi Photon tarjoaa globaalisti ylläpidetyt palvelimet (kuva 2), jotka varmistavat pelaajille nopeasti toimivat yhteydet. Photon pilvipalvelu osaa myös skaalata verkkokäyttäytymisen pelaajien määrän mukaan ja jakaa kapasiteettia tasaisesti. Lisäksi palvelu on aloittaville pelinkehittäjille ilmainen. Rajoittavana tekijänä on mm. 20 samanaikainen pelaaminen heidän palvelussaan. Tarvittaessa kehittäjät voivat päivittää tilaustaan useamman pelaajan palveluun ja suhteellisen suopeaan hintaan. Ratkaisuista kerrotaan myöhemmin lisää.

Photon tarjoaa palveluita pienien peliyrityksien lisäksi todella suurille organisaatioille. Kotisivuilla mainostetaan noin 270 000 kehittäjän/studion käyttävän Exit Gamesin palveluita, ja isoimpina peliyrityksinä on mainittu mm. Square Enix sekä Codemasters, joiden pelien pelaajamäärät ovat huomattavasti suurempia. (Exit Games 2018a.)

Exit Gamesin tarjoamia palveluita voidaan soveltaa mihin tahansa pelimuotoon. Palvelu kykenee tarjoamaan nopean yhteyden niin neljän kuin useamman sadankin pelaajan välillä. Verkkopeliä kehittäessä tulee kuitenkin ottaa huomioon, minkä Photon-palvelun tuotteen valitsee, koska osa tuotteista hyödyntää mm. P2P-yhteyttä, mikä ei sovellu peleihin jossa on useita satoja pelaajia samaan aikaan. Mikäli kehittäjä kokee, että Photonin tarjoamat yhteydet eivät riitä, voi kehittäjä valita itse ylläpidettävän palvelinympäristön. Photon pystyy kuitenkin käsittelemään suuria pelaajamääriä tarvittaessa. Tästä esimerkkeinä ovat mm. MMORPG-pelit Albion Online ja Das Tal. (Exit Games 2018b.)

Kuva 2. Palvelinten sijainti. (Exit Games 2018e.)

3.1 Photon palvelun rakenne

Photon tarjoaa palvelua kahdessa eri muodossa: pilvipalveluna (SaaS) tai itse ylläpidettävänä palvelinratkaisuna, jossa asiakas ylläpitää omaa palvelinjärjestelmää. Kyseisessä ratkaisussa asiakas vastaa itse palveluiden ylläpidettävyydestä ja teknologiasta, jossa Photon-palvelua ajetaan. Tämä antaa tiettyjä rasitteita kuten palveluiden pystyttäminen ja ylläpitomaksut voivat olla loppujen lopuksi paljon kalliimpia kuin palvelu tarjottuna olisi. Itse ylläpidettävyydessä kehittäjä saa kuitenkin täyden vapauden luoda itse yhteydet ja muokata palvelinpään ohjelmistoa haluamakseen. (Exit Games 2018c.) Exit Gamesin ylläpitämät Photon tuotteet toimivat kaikki saman pääsovelluksen alla, jota kutsutaan Photon Cloudiksi (kuva 3).

Kuva 3. Palvelinrakenne. (Exit Games 2018b.)

3.2 Palvelutarjonta

Exit Gamesin tarjoama tuoteperhe koostuu kolmesta isommasta kokonaisuudesta: moninpeli, kommunikaatio sekä self-hosted-palvelumuoto. Moninpelikokonaisuus sisältää kolme eri teknologialla ja arkkitehtuurilla toimivaa palvelua, ja ne eroavat hinnoiltaan ja ominaisuuksiltaan. Kaikki kolme moninpelipalvelua käyttävät Photon ”Realtime”-toiminnallisuutta, joka tarjoaa saumattoman yhteyden globaalisti samalla alueella pelaavien pelaajien välille. Kaikki tuotteet mahdollistavat eri alustoilla pelaamisen keskenään. Esimerkiksi kehittäjien luomaa peliä voi pelata PC- sekä Android-pohjaisilla laitteilla.

3.3 Photon Quantum ja deterministinen prosessointi

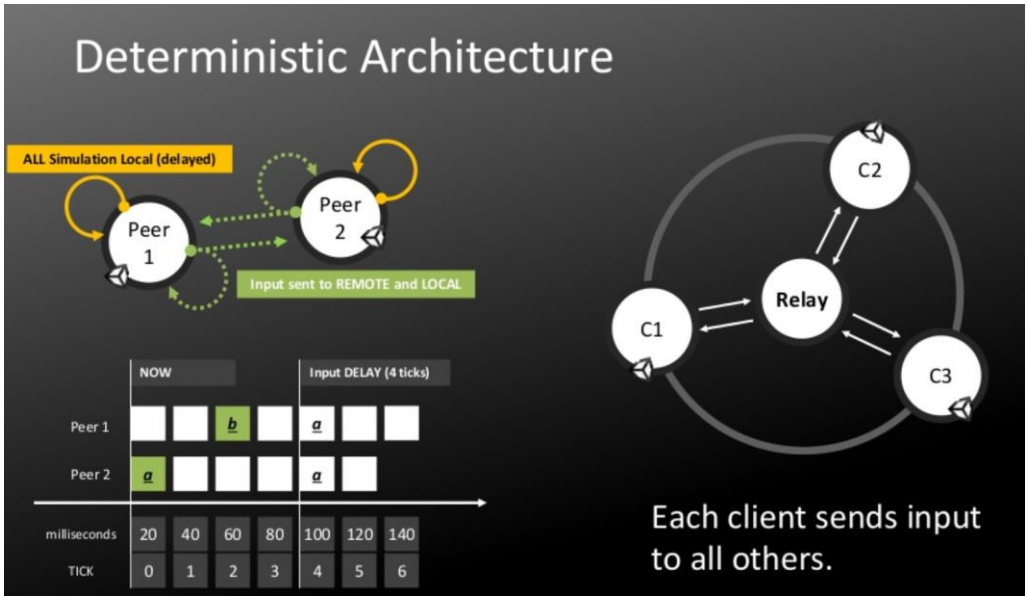
Quantum on Photon tuoteperheen kallein ja edistynein verkkopalveluteknologia. Quantum on luotu ja suunniteltu nopeatempoisille reaaliaikaisille peleille kuten mm. MOBA, RTS, brawler-pelit, urheilupelit ja tappelupelit. Quantum käyttää determinististä arkkitehtuuria. Tämä tarkoittaa sitä, että kaikki pelaajien painallukset ja liikkeet tallennetaan palvelimelle ja ne lähetetään kaikille pelaajille samaan aikaan takaisin (kuva 4). Sen sijaan että pelaajille lähetettäisiin tieto pelaajien eri tiloista, lähetetään pelaajille tieto syötteistä, joiden avulla esimerkiksi pelaajien liikkeet voidaan toistaa. (Wegmann 2017.)

Kyseinen toimintamalli ei vaadi kovinkaan suuria kaistoja lähettää tietoa eteenpäin, sekä sen avulla on helppo toteuttaa uudelleentoistot esimerkiksi tilanteista, mitä aikaisemmassa pelissä on tapahtunut. Koska kaikki syötteet tallennetaan palvelimen puolelle, voidaan helposti toistaa, missä ja mitä eri pelaajat ovat tehneet. Deterministinen prosessointi mahdollistaa myös ison määrän eri liikkuvia osia verkon ylitse. (Wegmann 2017.)

Kyseinen prosessointi aiheuttaa kuitenkin haasteita. Se on hankala implementoida ja huonon verkkoyhteyden omistava pelaaja aiheuttaa paljon viivettä. (Wegmann 2017.)

Quantum on tällä hetkellä yksi kehittyneimmistä arkkitehtuureista ja mahdollistaa esimerkiksi eri maista tulevien pelaajien keskinäisen toiminnan verkon ylitse. Lähteen linkissä on video tilanteessa, missä toinen pelaaja pelaa Brasiliassa ja toinen Ruotsissa. Vaikka pelaajien väliset yhteydet palvelimeen ovat eritasoiset, tapahtuvat liikkeet ja ammusten lentäminen/osuminen täysin samaan aikaan molemmilla yksilöillä. (Robert 2017.)

Quantum on hintaluokaltaan todella kallis kahteen muuhun verkkoteknologiaan ja kustantaa vuodessa 10,000 dollaria. Hinta sisältää täyden SDK-hallinnan ja kaiken mahdollisen avustuksen Quantum tekijöiltä. (Exit Games 2018f.)



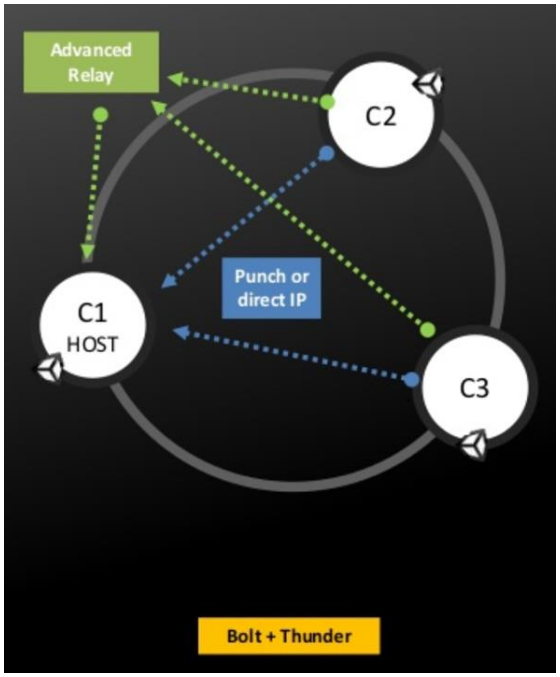
Kuva 4. Deterministinen arkkitehtuuri. (Wegmann 2017.)

3.4 Photon Bolt

Photon Bolt hyödyntää P2P-verkkotoiminnallisuutta, joka tarkoittaa, että se hyödyntää tiedonsiirrossa pelaajien yhteyksiä toisiinsa. Bolt-tuotteen ominaisuutena on, että pelaaja, joka luo pelin, toimii ns. palvelimena. Kaikki tieto, joka siirtyy pelaajilta toisille, liikkuu kyseisen pelaajan luoman pelin ylitse (kuva 5). Toimintamalli luo myös edun tietyissä peleissä pelaajalle, joka on luonut pelin. Tämä johtuu siitä, että kyseinen pelaaja näkee kaikki muutokset heti. Muut pelaajat saavat kaiken mahdollisen tiedon millisekunteja myöhemmin. Pelin ylläpitäjän hyöty ei ole kuitenkaan suuri ja rajoittuu vain tietynlaisiin peleihin. Esimerkiksi FPS-peleissä pelaaja näkee kulman takaa tulevan pelaajan ennemmin kuin pelaaja, joka tulee kulman takaa. (Exit Games 2018d.)

Bolt- ja P2P-yhteys tuo paljon hyötyjäkin. P2P-yhteyden avulla Bolt luo pelaajien välille nopeat yhteydet ja se osaa välttellä yleisiä P2P:n aiheuttamia haittoja kuten huonoimman yhteyden omaava pelaaja asettaa viiveen muille pelaajille. Photon Bolt on luotu huonepohjaisiin peleihin ja toimii useiden kymmenien pelaajien kesken. Peli ei kuitenkaan sovellu suurien pelikokonaisuuksien ajamiseen, jolloin pelin tekijän internetyhteys on pullonkaulana. (Wegmann 2017.) Bolt on hinnoittelultaan samanhintainen kuin PUN, josta kerrotaan lisää seuraavassa osuudessa. (Exit Games 2018d.)

Bolt tarjoaa myös linkittävän yhteyden (advanced relay), mikäli pelaaja ei jostain syystä saa yhteyttä suoraan pelin ylläpitäjään. Tällöin pelaaja käyttää hyödykseen Photon Cloud -palvelua ja auttaa luomaan yhteyden pelin ylläpitäjään (kuva 5).



Kuva 5. Photon Bolt arkkitehtuuri. (Wegmann 2017.)

3.5 Photon PUN

Photon PUN (Photon Unity Networking) on verkkotyökalu, joka voidaan implementoida suoraan Unity-pelimoottorin projektiin. API sisältää useita eri valmiita toiminnallisuuksia ja helpottaa kehittäjän verkkomoninpelin kehittämistä runsaasti.

PUN käyttää muiden tuotteiden tapaan Photon Cloud- ja Realtime-ominaisuuksia. Bolt tuotteeseen verrattuna PUN ei hyödynnä pelaajien konett ja verkkoa vaan pyörii täysin Exit Gamesin ylläpitämillä palvelimilla, mikäli kehittäjä ei ole valinnut Photon Server-vaihtoehtoa. Toimintamalli on ”Client to Server”, ja se hyödyntää UDP-, TCP-, http- ja WebSocket-teknologioita. UDP, TCP ja WebSocket ovat verkossa tiedonsiirtoon käytettyjä tietoliikenneprotokollia. UDP on esimerkiksi paljon nopeampi, mutta epävakaampi protokolla kuin TCP. TCP tarkistaa aina, että verkon ylitse siirretty paketti on saapunut vastaanottajalleen, kun taas UDP ei edes luo yhteyttä laitteiden välille. PUN käyttää myös ”Advanced Relay”-arkkitehtuuria (kuva 6), missä kaikki pelaajat yhdistävät

”huoneisiin”. Huone voi toimia kevyenä palvelimena tai puhtaasti vain ns. viestin välittäjänä. (Exit Games 2018e.)

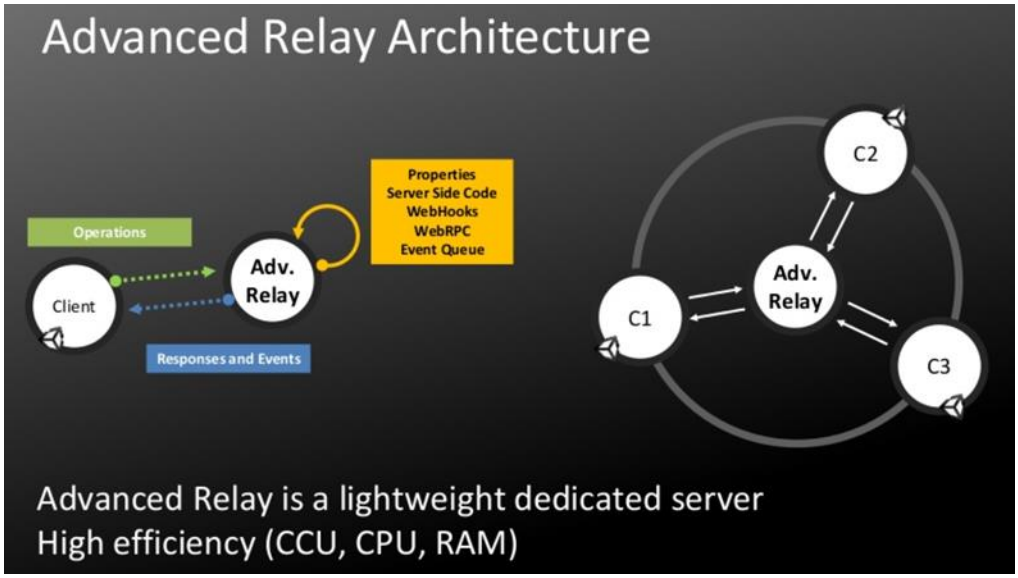
Kyseinen työkalu mahdollistaa nopeat verkkoyhteydet pelaajien kesken etsimällä parhaimman palvelimen aluetoiminnallisuuden vuoksi. Kyseinen toiminto etsii automaattisesti palvelimen mihin pelaajalla on nopein yhteys. Kuitenkaan eri mantereiden välillä pelaavien pelaajien yhteys ei ole paras mahdollinen, sillä pelaaja, joka tekee huoneen määrittää, mitä palvelinta käytetään yhteyspisteenä.

Tuote on aloittaville verkkomoninpelien kehittäjille hyvä vaihtoehto, koska Photon tarjoaa ilmaiseksi 20 yhdenaikaisen käyttäjän tilauksen ilmaiseksi. PUN-työkalun asentaminen on helppoa, ja sen implementointi omaan projektiin on suhteellisen vaivatonta alkuselvityksien ja opiskelun jälkeen. Lisäksi Photonin backend tekee kehittäjien puolesta paljon asioita. PUN API ja dokumentaatio helpottavat käyttöönottoa ja sen yhdistämistä oman pelin koodipohjaan.

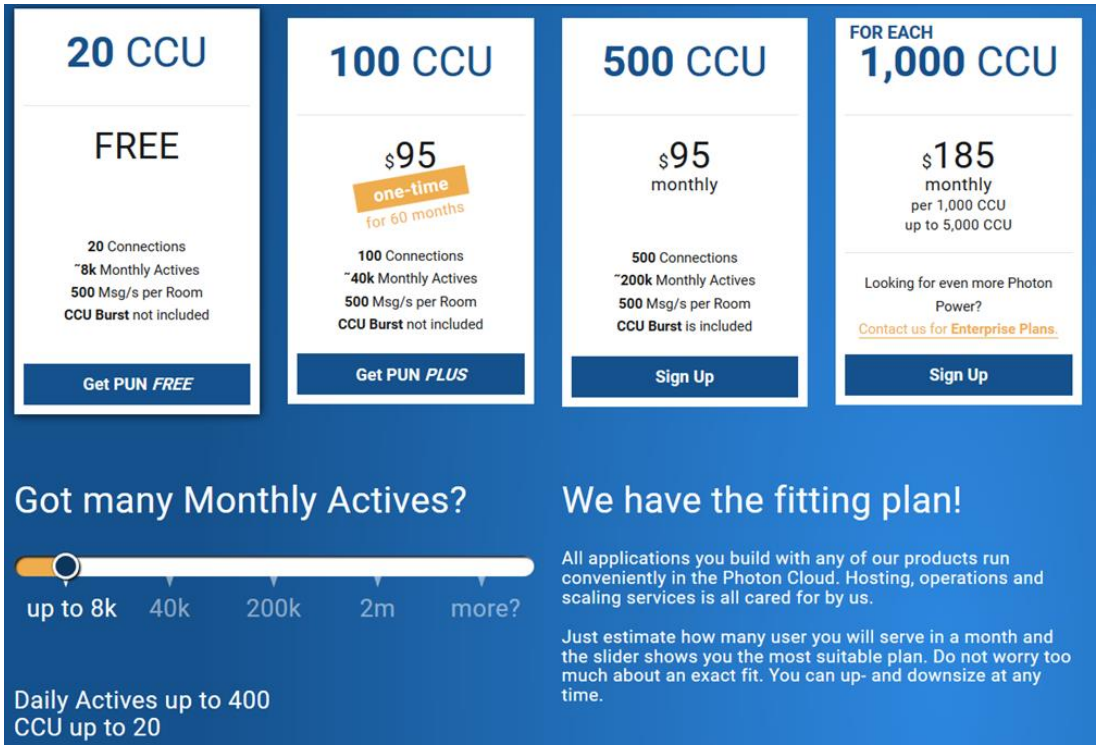
PUN tuo mukanaan myös ”matchmaking”-ominaisuuden, mikä mahdollistaa helposti satunnaisten pelaajien keskenään pelaamisen. Photon keskittyy huonepainoitteiseen moninpelaamismalliin, missä pelaajat luovat pelihuoneita, johon pelaajat voivat liittyä. (Exit Games 2018e.)

Ratkaisumalli tuo myös mukanaan paljon helpottavia ominaisuuksia kuten helpot takaisinkutsukomennot, helposti synkronoitavat komponentit peliobjekteihin sekä etäkutsut, joiden avulla voidaan esimerkiksi tuoda jokaisen pelaajan tietoon samaan aikaan tietty tapahtuma. Exit Games on myös julkaissut kyseisen työkalun lähdekoodin avoimena, mikä mahdollistaa jokaisen kehittäjän muokata työkalua. Avoimen lähdekoodin lisäksi kotisivuilta löytyy useita helpottavia demoja ja oppaita, joiden avulla kehittäjät pääsevät alkuun. (Exit Games 2018e.)

PUN-työkalusta löytyy kaksi eri versiota. Toinen versio on ilmainen, josta aikaisemmin on mainittu. Toinen versio maksaa 95 dollaria ja sisältää 100 yhdenaikaisen pelaajan sopimuksen 60 kuukauden ajaksi. Mikäli peli menestyy ja saa lisää aktiivisia pelaajia, voi pelinkehittäjä päivittää sopimustaan niin laajaksi kuin tarve vaatii (kuva 7). (Exit Games 2018g.)



Kuva 6. PUN Advanced Relay arkkitehtuuri. (Wegmann 2017.)



Kuva 7. Photon Bolt ja PUN hinnoittelu. (Exit Games 2018g)

3.6 Verkkoalustan valinta

Verkkomoninpelialustan valinnan kriteerejä Firetail Gamesille olivat hinnoittelu, helppokäyttöisyys, palvelimien ylläpito ja se, millä pohjalla alusta toimii ja pyörii. Photon tarjoaa paljon skaalautuvuutta ja mahdollisuuden parantaa sopimusta ja lisäämällä tilaa enemmän yhdenaikaisille pelaajille. Photon on myös hinnoittelultaan sopiva startup-peliyritykselle tuodessaan vaihtoehdon aloittaa ilmaisella sopimuksella. Mikäli julkaistun pelin suosio kasvaa, voidaan pelaajamäärää kasvattaa.

Photon hoitaa myös palvelinten ylläpitämisen ympäri maailmaa ja kuten aikaisemmin mainittu, asiakas voi valita myös itse ylläpidettävän vaihtoehdon. Photon on myös erittäin soveltuva Unity-pelimoottorille, jota FTG käyttää pelin kehittämiseen.

Photon alustaan päätyminen vaikutti myös aiempi kokemus ja Nordic Games messutapahtumassa käytyjen tapaamisten johdosta. Vaihtoehtoina yrityksellä oli Amazon, XtraLife sekä Unity-pelimoottorin oma verkkotyökalu Unet. Taulukossa 1 on vertailu kyseisten vaihtoehtojen välillä. Photon puolelta on vain otettu vertailuun PUN, koska se soveltui parhaiten yrityksen peliin Boltin ja Quantumin sijaan.

Taulukko 1. Vaihtoehtoiset verkkoalustat.

	Hinnoittelu	Helppokäyttöisyys	Palvelintarjonta	Lisätietoa
Photon PUN	Ilmainen aluksi ja helposti skaalautuva. Hinta 0-185$ tai enemmän.	Helposti asennettava lisäosa. Dokumentaatio ja oppaat todella laajasti saatavilla.	Kyllä	Tarjoaa useita eri vaihtoehtoja pelaajien yhdistämiseen keskenään.
Unet	Ilmainen 20 käyttäjälle samaan aikaan. Hinta 0-125$ tai enemmän. Laskuttaa myös siirretystä datasta.	Löytyy valmiiksi Unity-pelimoottorista. Paljon dokumentaatiota ja oppaita saatavilla.	Kyllä	Ei tarjoa ”relay” palvelua vaan pitää ostaa erikseen.
Amazon	Laskuttaa siirretyn datan mukaan, hinta vaihtelee paljon.	Vaihtelevasti ohjeita ja paljon mobiilipuolelle. Ei suoraan asennettavaa lisäosaa.	Kyllä	Laaja, skaalautuva, mutta haasteellinen yhdistää omaan projektiin.
XtraLife	Ilmainen, lisää ominaisuuksia premium-palvelulla. Ei tietoa hinnasta.	Saatavilla olevan dokumentaation taso kohtalainen, vaikka kyseessä avoimen lähdekoodin työkalu.	Kyllä	Sitoo paljon ominaisuuksia premium-palvelun taakse.

4 Moninpelin suunnittelu

Firetail Gamesin luoma peli on nopeatempoinen fysiikkapohjainen “brawler”-peli. Pelissä voi samaan aikaan pelata maksimissaan neljä pelaajaa tai mikäli pelaajia puuttuu, voidaan tuoda tekoälyllisiä pelaajia peliin mukaan. Tekoälyllisten pelaajien tuominen verkkomoninpeliin ei ollut kuitenkaan ajankohtainen opinnäytetyön aikana ja niiden toteuttaminen jätettiin myöhemmälle.

Alkuperäinen peli oli luotu niin, että sitä pelataan ainoastaan paikallisesti, eikä sen kehitysvaiheessa otettu huomioon verkkomoninpeliin liittyviä asioita. Vaikka paikallinen moninpeli on hyvä pohja lähteä toteuttamaan verkkomoninpeliä, alkuperäinen pohja pelille on sekavasti toteutettu ja sen uutta pohjaa toteutetaan opinnäytetyön aikana.

Pelin nopeatempoisuus tuo paljon haasteita verkkomoninpeli-ominaisuutta ajatellen. Paljon samaan aikaan tapahtuvia liikkeitä pitää tarkkailla jatkuvasti ja niiden pitää olla ajan tasalla kaikilla pelaajilla samaan aikaan. Pienikin viive voi aiheuttaa lyönnin tai ammuksen ohi menemisen, jolloin toinen pelaaja saa edun tilanteessa.

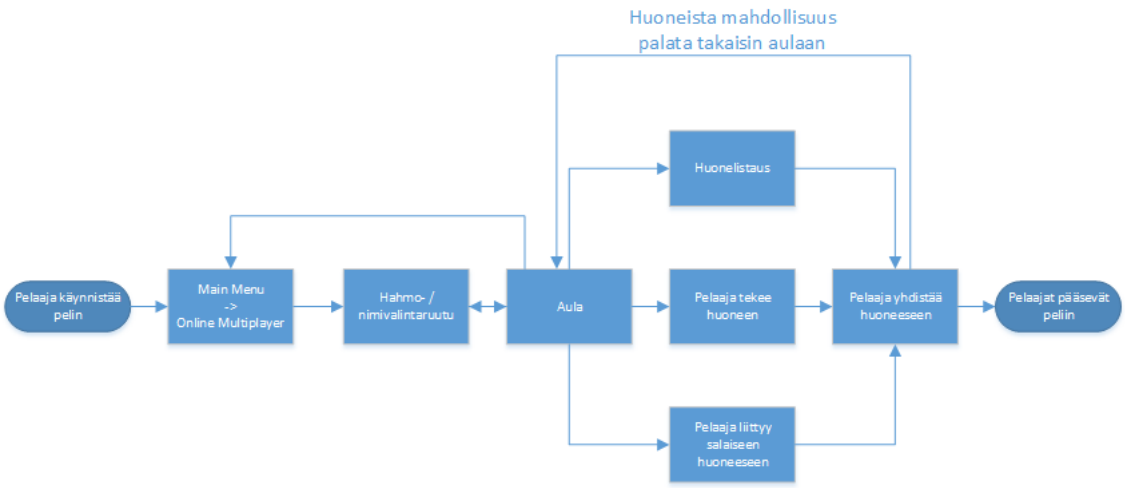
Verkkomoninpelissä on tärkeää ottaa siis huomioon, että viive on mahdollisimman vähäinen. Viivettä voidaan parantaa pelaajien kesken ostamalla parempia verkkopalveluita, rajoittamalla miten lähellä maantieteellisesti pelaajat ovat lähellä toisiaan ja optimoimalla verkkoliikennettä mahdollisimman vähäiseksi. Koska PUN ja Photon Cloud ovat Exit Gamesin tarjoama palvelu, tulee tietää, missä heidän palvelimet sijaitsevat. PUN työkalu onneksi valitsee automaattisesti lähimmän palvelimen pelaajan asuinalueen mukaan. Kuten aikaisemmin on mainittu, palvelimia sijaitsee 11 eri maassa ympäri maailmaa.

Kun pelin verkkomoninpeliä kehitetään, on otettava viiveen lisäksi paljon muitakin asioita huomioon. Tieto pelaajista ja heidän liikkeistään, tieto ammuksista ja niiden omistajista ovat tärkeintä ja priorisoitua tietoa, mitä tulee siirtää internetin ylitse. Kanssapelaajien on myös tärkeä nähdä toisten pelaajien animaatiot, jotta pelaajat pystyvät ennakoimaan muiden pelaajien tekemisiä. Muutamissa pelin aseissa on ns. latausmekanismi ja niiden väistäminen on helpompaa, kun se voidaan ennakoida.

Pelaajien liikkeiden seuraamisen lisäksi on seurattava pelissä tapahtuvia muutoksia, jotka lähetetään samaan aikaan kaikille pelaajille. Esimerkiksi Jokaisen uuden pelin latauksen yhteydessä kenttä tulee ladata kaikille samaan aikaan. Myös pelissä kerättävien aselaatikoiden tulee olla kaikille samaan aikaan saatavilla. Aselaatikot häviävät, kun ne poimitaan ja ne luodaan uudestaan tietyn ajan kuluttua. Pelaajien tulee myös tietää, mikä on heidän välinen pistetilanne ja milloin peli on päättynyt.

4.1 Moninpelin vaiheistuksen suunnittelu

Moninpelin vaiheistuksen suunnittelussa päädyttiin suoraviivaiseen ratkaisumalliin (kuva 8). Ratkaisumallissa pyrittiin luomaan käyttäjäystävällinen ja suoraviivainen kokonaisuus. Pelin käynnistyessä pelaaja voi valita Online-valikon, josta avautuu suoraan pelaajalle hahmovalinta sekä nimen asettaminen. Valintojen jälkeen pelaaja pääsee aulaan, josta voidaan luoda peli/huone, palata takaisin päävalikkoon/hahmovalintaan, nähdä avoinna olevat pelit sekä liittyä joko avoimeen tai salaiseen peliin. Kyseisessä ratkaisumallissa pelaajat eivät näe käynnissä olevia pelejä.



Kuva 8. Moninpelin vaiheistus.

4.2 Hahmon ja nimen asettaminen

Yksi vaikeimmista toteutettavista asioista suunnitteluvaiheessa oli päättää missä vaiheessa pelaajat tekevät hahmovalinnan. Toteutuksessa päädyttiin kuitenkin siihen, että hahmovalintaruutu sijoitetaan ensimmäiseksi vaiheeksi verkkomoninpelissä.

Hahmovalinnassa valitaan haluttu pelihahmo sekä nimi, joka näkyy pelissä oman hahmon yläpuolella. Valittu hahmo ja nimi tallennetaan pelaajan kiintolevylle, joten peli muistaa ne automaattisesti seuraavalla kerralla. Tehokkain tapa olisi ollut ottaa nimi suoraan Steam-palvelusta, mutta Steamin yhdistäminen peliprojektiin koettiin liian suureksi työksi kaiken muun ohella tässä vaiheessa.

Yksi vaihtoehto hahmovalinnalle olisi sijoittaa se suoraan odotushuoneeseen. Jokainen voisi valita hahmon menemällä “Change Character”-valikkoon ja valita yhden hahmon kuudesta. Jokaista hahmoa voi olla vain yksi kerrallaan pelissä. Suunnitteluvaiheessa todettiin kuitenkin, että kuluisi liian paljon aikaa ennen kuin peli voitaisiin käynnistää. Peliin pääsemisestä pyrittiin luomaan mahdollisimman nopea ja suoraviivainen, johon tämä toteutustapa ei sopinut.

Toinen vaihtoehto hahmovalinnalle oli toteuttaa Overwatch-pelissä käytetty valintavaihe, kun peli alkaa. Kun peli on käynnistetty huoneesta, on pelaajilla 20 sekuntia aikaa valita hahmo tai se valitaan pelaajalle automaattisesti. Kun jokainen on valinnut hahmonsa, peli alkaa tietyn ajan jälkeen. Tämäkin toteutustapa todettiin huonoksi, koska se veisi liikaa aikaa pelaajilta.

Hahmovalinta ei vaikuta pelin toiminnollisiin ominaisuuksiin kuten pelaajan nopeuteen tai kestävyyteen. Hahmovalinta vaikuttaa vain siihen, että pelaajat erottuvat toisistaan ja pelaaja tietää mitä hahmoa ohjataan. Tämän vuoksi pelaajavalinta on sama tehdä jo ennen kuin pelaaja edes pääsee liittymään huoneisiin.

Käytetyn toteutustavan huonoin puoli on siinä, että pelaajat voivat olla samannäköisiä, koska kaikilla on mahdollisuus valita sama hahmo, tietämättä siitä, mitä hahmoa muut pelaajat käyttävät. Kyseinen ongelma kuitenkin tiedostettiin ja jokaisen pelaajan yläpuolelle luodaan pelaajaa ja hahmoa yhdistävä nimi-kenttä.

4.3 Aula

Hahmovalinnan jälkeen pelaaja siirtyy aulaan. Aulan tarkoituksena on tuoda pelaajat yhteen tilaan, josta voidaan luoda huoneita tai liittyä huoneisiin. Pelaaja pääsee aulaan päänäkymän kautta.

Aulan tulisi olla mahdollisimman yksinkertainen ja helposti lähestyttävä. Aulassa ei tulisi olla muuta kuin lista auki olevista huoneista ja muutama painike. Painikkeina on paluu takaisin päänäkymään tai hahmovalintaan, huoneen luonti sekä liittyminen yksityiseen huoneeseen. Huoneen luomispainikkeen takaa löytyy muutama tekstikenttä, johon voidaan asettaa huoneen nimi, onko huone salainen ja sekä maksimipelaajamäärä.

Huoneeseen liittyminen tapahtuisi yksinkertaisesti niin, että kun huone on listattuna, sitä painamalla pelaaja yhdistää joko huoneeseen tai epäonnistuu huoneeseen liittymisessä. Tästä ilmoitettaisiin pelaajalle, miksi yhdistäminen epäonnistuu.

4.4 Huoneet

Huoneen tarkoituksena on yhdistää pelaajat samaan peliin. Aulan kautta pelaajat voivat liittyä samaan huoneeseen, josta voidaan käynnistää yhteinen pelisessio. Mikäli huoneen tekijä on salannut huoneen, voidaan salattuun huoneeseen liittyä ”Join private game”-painikkeella. Painike tuo esille tekstikentän, johon kirjoitetaan luodun pelin nimi.

Huoneen luoja voi käynnistää pelin, potkia muita pelaajia pois tai vaihtaa kentän, millä peli käynnistetään. Huoneessa jokaisella pelaajalla on valmius-painike, jolla voitaisiin indikoida muille pelaajille, että pelaajat ovat valmiita. Huone näkymässä pelaajat näkevät toisensa ja voivat poistua milloin tahansa. Huone kaatuu, kun huoneen tekijä lähtee pois.

5 Toteutus

Ennen toteutuksen aloitusta kävin läpi useita eri lähteitä ja otin selvää, kuinka eri tavoin PUN-työkalua voidaan hyödyntää. Verkosta löytyy paljon erilaisia oppaita, keskusteluja sekä videoita eri toiminnollisuuksien toteutusta varten.

Ensimmäiseksi lähdin havainnollistamaan, kuinka esimerkkidemon kautta on luotu yhteys Photon-palvelimiin ja kuinka tieto välitetään pelaajalta toiselle. Photonin kotisivuilta löytyy useampi Unity-projekti ja -demo, joita kehittäjät voivat käyttää referenssinä omiin projekteihin.

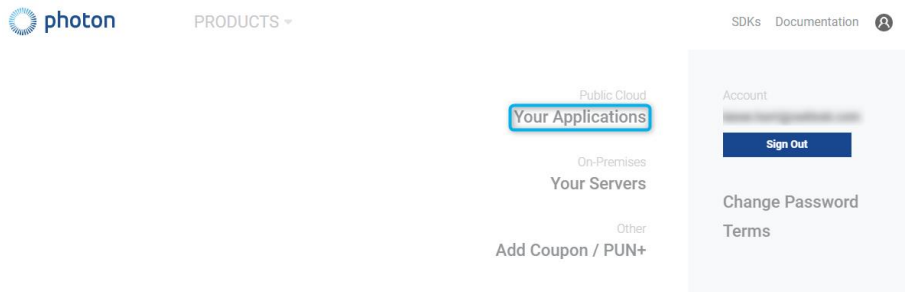
Opinnäytetyön aikana tehty toteutus on ns. raakaversio miltä itse moninpeli tulee tulevaisuudessa näyttämään. Ulkoasultaan kokonaisuus tulee näyttäämään paljon viimeistellymmältä sekä siihen tuodaan lisää ominaisuuksia. Toteutuksessa otettiin mallia esimerkiksi ”First Gear Games” Youtube-kanavan video-opasteista. Videot olivat kuitenkin vain suuntaa antavina ja auttoivat kehittämisessä.

Toteutuksessa on näytetty vain palasia lähdekoodista, koska koko lähdekoodin näyttäminen avaisi hackereille/huijaajille mahdollisuuden tehdä ohjelmia, joilla he saisivat edun muihin pelaajiin. Huijausohjelmat voivat esimerkiksi antaa pelaajalle kuolemattomuuden tai loputtomat panokset.

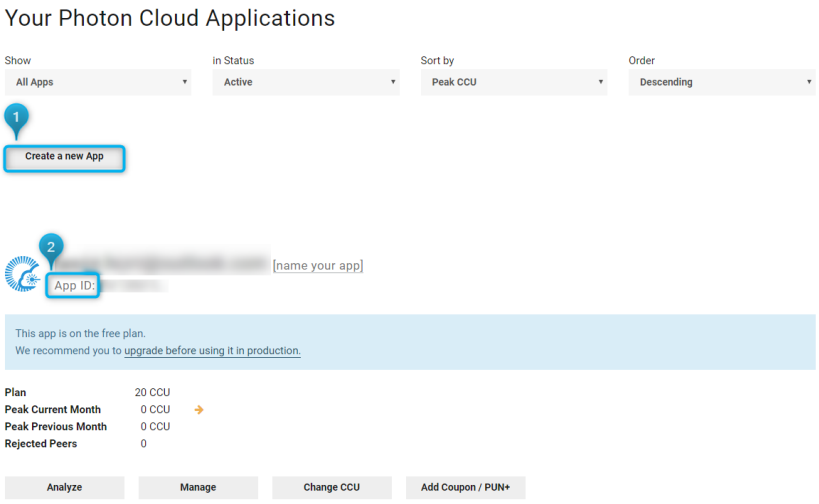
5.1 PUN asentaminen ja konfigurointi

PUN-työkalu on yksinkertainen asentaa ja konfiguroida. Kehittäjän tulee ensimmäiseksi rekisteröidä tili Photonin kotisivuilla. Tilin avulla kehittäjä voi hallinnoida palvelimelle luotuja applikaatioita. Applikaatiot määrittävät mihin palvelua käytetään ja esimerkiksi seuraamaan, kuinka monta pelaajaa käy pelaamassa peliä. Tilin avulla autentikoidaan Unityn puolella, mihin palvelimeen ja applikaatioon peli yhdistää käynnistyksen yhteydessä. Tilin luominen on suoraviivaista eikä vaadi muuta kuin käyttäjän perustiedot.

Tilin luomisen jälkeen kehittäjän tulee kirjautua palveluun ja luoda uusi applikaatio palveluun. Applikaation luominen tapahtuu menemällä “Your applications”-valikkoon klikkaamalla profiilikuvaa oikeasta yläkulmasta (kuva 9). Mikäli sivustolla näkyy luotuja, käyttämättömiä applikaatioita, voidaan niitä käyttää. Uuden applikaation luonti tapahtuu klikkaamalla “Create new app”-painiketta (kuva 10) sivuston ylälaidassa. Käyttäjältä kysytään mitä Photon palvelutyyppiä käytetään, mikä sen nimeksi/kuvaukseksi asetetaan sekä mahdollinen URL-osoite. Luonti viimeistellään painamalla “Create”-painiketta.



Kuva 9. Photon käyttäjävalikko.

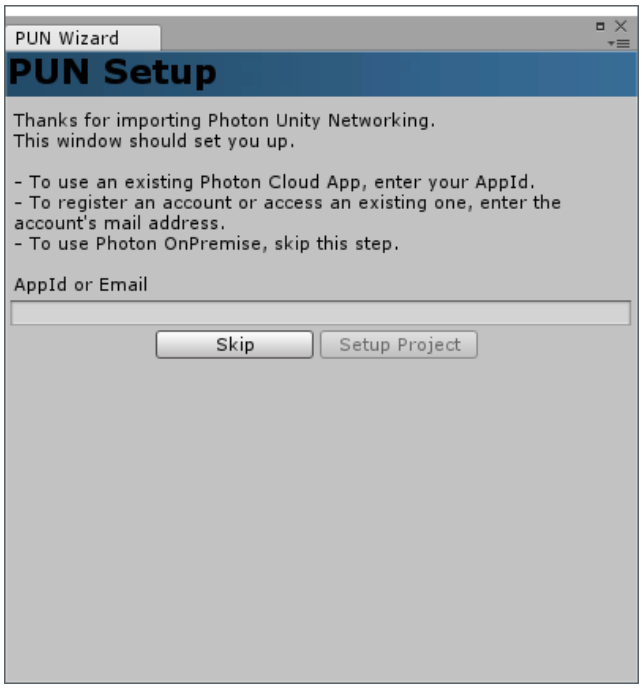


Kuva 10. Photon applikaatiot.

Kun uusi applikaatio on luotu, sitä voidaan tarkastella “Your applications”-näkymästä tarkemmin. Applikaatiosta ja sen tilasta saadaan paljon tietoa, mutta tässä vaiheessa tarvitaan vain sen “App ID”. App ID on mikä luodaan applikaatiolle automaattisesti sen luonnin yhteydessä ja sen avulla tapahtuu applikaatioon yhdistäminen (kuva 10).

Tunnuksen ja applikaation luomisen jälkeen luodaan (tai avataan vanha) Unity-projekti. Kun projekti on auki, avataan Unitystä “Asset Store”-verkkokauppa ja etsitään Photon Unity Networking-työkalu ja ladataan se. Kun työkalua tuodaan projektiin kannattaa valita vain kaikki tarvittava. Työkalu esimerkiksi sisältää demoja ja oppaita, joita ei välttämättä enää tarvita kehitysvaiheessa. Demot ja oppaat voidaan poistaa myös myöhemmässä vaiheessa.

Kun työkalu on tuotu Unity-ohjelmistoon, avautuu PUN Wizard-asennusohjelma (kuva 11), jossa määritetään mihin applikaatioon otetaan yhteys. Tässä vaiheessa tarvitaan aikaisemmin luodun palvelun App ID-tunnusta. Kun ID on syötetty, painetaan “Setup Project”. Asennus kestää muutaman minuutin. Asennus asettaa automaattisesti käyttäjälle parhaiten soveltuvimmat asetukset. Asennus luo myös “PhotonServerSettings”-tiedoston Unityn-tiedostohierarkiaan, mistä voidaan myöhemmin muokata asetuksia. Asetuksista voidaan esimerkiksi muokata palvelutyyppi, käytetyn palvelimen alue ja mitä verkkoprotokollaa käytetään tiedonsiirtoon. Asennusvaiheen jälkeen voidaan aloittaa itse verkkopelin toteuttaminen. (Exit Games 2018i.)



Kuva 11. PUN Wizard.

5.2 Yhdistäminen palvelimeen ja perus asetusten säätäminen

Kun PUN on asennettu, tarvitsee käyttäjän vielä erikseen yhdistää Photonin pääpalvelimeen (master server), sekä palveluun missä ylläpidetään pelin verkkoa. Kun käyttäjä on yhdistänyt palveluun, käyttäjä liittyy aulaan, josta voidaan siirtyä eri huoneisiin.

Palvelinrakennetta on helppo ajatella hotellina. Pääpalvelin on katu, missä on hotelleja, palvelu on yksi tietty hotelli, jonne päästään tietystä ovesta. Hotellin aula ja huoneet kuvaavat pelissä näkyvää aulaa ja huoneita. Aulasta voidaan päästä toisiin huoneisiin ja takaisin. Mikäli huoneet ovat suljettuja, tarvitsee käyttäjä sinne päästäkseen avaimen.

Järkevintä yhteyden ylläpitämiseen on luoda peliobjekti, joka ei tuhoudu missään vaiheessa, kun pelaaja on verkkopeli-tilassa. Peliobjekti sisältää skriptin, mikä luo ja ylläpitää yhteyttä ja reagoi muutoksiin tiettyihin muutoksiin.

Yhdistäminen PUN-verkkoon tapahtuu yksinkertaisella koodilla, joka suoritetaan online-valikkoon mentäessä (kuva 12). (Exit Games 2018i.)

```
// Use this for initialization
void Start () {
    if (!PhotonNetwork.connected)
    {
        print("Connecting to server");
        PhotonNetwork.ConnectUsingSettings("0.1");
    }

    DontDestroyOnLoad(gameObject);
}
```

Kuva 12. Palvelimeen yhdistäminen.

Versionumero voi olla mikä tahansa kehittäjän asettama. Versionumeron tarkoitus on erottaa peliversiot toisistaan. Mikäli esimerkiksi käyttäjillä on eroava versionumero, heille voidaan ilmoittaa, että käytössä on esimerkiksi vanhentunut versio pelistä.

Kyseinen komento käyttää aikaisemmin luotuja asetuksia asennusvaiheessa. Asetuksia voidaan muokata PhotonServerSettings-komponenttia muokkaamalla. Mikäli kehittäjä

käyttää omaa palvelinta ylläpitääkseen PUN-verkkoa, voi yhdistämiseen tarvittavat tiedot lähettää Connect()-komentoa käyttäen.

ConnectUsingSettings-komento yhdistää pelaajan oikeaan palvelimeen ja applikaatioon. Käyttäjän tulee vielä liittyä aulaan liittyäkseen eri huoneisiin. Aulaan voidaan liittyä joko automaattisesti tai komennolla. Mikäli kehittäjä haluaa luoda automaattisen pelin hakemisen, tulee automaattinen aulaan liittyminen ottaa pois päältä. Aulaan liittyminen on oletuksena pois päältä. Koska haluamme luoda omia huoneita ja että pelaajat voivat liittyä huoneisiin, pelaajien tulee päästä ensimmäiseksi aulaan. Aulaan yhdistäminen toteutetaan kutsumalla metodia *PhotonNetwork.JoinLobby()* (kuva 13). Metodia kutsutaan, kun pelaaja on yhdistänyt master-palvelimeen. *OnConnectedToMaster*-metodi on Photonin sisältämä metodi, joka tarkkailee, milloin pelaaja yhdistyy palvelimeen. PUN-työkalusta löytyy useita valmiita toiminnallisuuksia, joiden avulla voidaan seurata eri tilanteissa pelaajien toimintoja. Työkalun avulla voidaan esimerkiksi helposti seurata, milloin pelaaja on yhdistänyt tai poistunut huoneesta *OnJoinedRoom*- ja *OnLeftRoom*-metodien avulla. (Exit Games 2018i.)

```
public override void OnConnectedToMaster()
{
    print("Connected to master");
    PhotonNetwork.automaticallySyncScene = true;
    PhotonNetwork.JoinLobby(TypedLobby.Default);
}
```

Kuva 13. Aulaan liittyminen.

5.3 Toiminnollisuuksien toteutus

5.3.1 Graafinen käyttöliittymä

Graafiseen käyttöliittymän toteutukseen käytettiin Unityn omia User Interface-komponentteja (UI) sekä Text Mesh Pro-lisäosan (TMP) tekstityökalua. TMP-tekstityökalua käytettiin sen monipuolisuuden vuoksi. Työkalulla voidaan muokata enemmän tekstin ulkoasua ja on käytettävyydeltään selkeämpi kuin Unityn sisältämä tekstityökalu.

Käyttöliittymä toteutettiin käyttämällä yhtä Unity-näkymää (Unity Scene). Mikäli käytettäisiin useampaa näkymää, käyttökokemus hidastuisi runsaasti ja olisi hankalampi toteuttaa. Tämä johtuisi siitä, että uusien näkymien latausajat voivat olla pitkiä ja pitkät latausajat pilaavat pelaajan käyttökokemuksen. Uuden näkymän vaihtuessa, kaikki peliobjektit pitäisi ladata uudestaan tai olla tuhoamatta niitä tietyn näkymän vaihtuessa, mikä ei ole järkevä toteutustapa. Peliobjektien uudestaan lataaminen on raskasta ja aiheuttaisi näkyvää viivettä objektien ilmestymisessä. Peliobjektien säilyttäminen myös voi aiheuttaa haitallisia tilanteita, jossa tietyt ylläpidettävät tiedot aiheuttaisivat ristiriitaisuuksia. Esimerkiksi jos tietoja huoneeseen yhdistäneistä pelaajista pidettäisiin koko ajan yllä, näkyisivät samat pelaajat huoneessa, vaikka tieto ei pidä paikkaansa.

Toteutus oli helpompi toteuttaa niin, että useita komponentteja aktivoidaan/deaktivoidaan tietyn napin painalluksen jälkeen. Näin ollen kaikkien komponenttien tiedot ovat tallella sekä huoneen ja aulan välillä on nopea liikkua. Kyseinen toteutustapa myös nopeuttaa testaamista, kun kaikki komponentit ovat samaan aikaan esillä. Näkymien vaihtaminen useaan kertaan hidastaisi runsaasti työnkulkua.

Graafisesta käyttöliittymästä pyrittiin tekemään mahdollisimman käyttäjäystävällinen pelaajalle. Painikkeiden asettelun tulisi olla johdonmukainen ja looginen. Opinnäytetyötä tehdessä graafinen ulkoasu jäi kuitenkin vielä prototyypiksi ja ei ole lopullinen.

5.3.2 Pelaajan nimen asettaminen ja hahmovalinta

Pelaaja voi asettaa tai vaihtaa nimen moninpelin ensimmäisessä ruudussa. Pelaajan nimi tallennetaan pelin muistiin, jotta pelaajan ei tarvitse joka kerta kirjoittaa nimeä uudestaan. Syötön yhteydessä pelaajan asettama nimi asetetaan Photon-verkon pelaajanimeksi. *SetPlayerName*-metodi on asetettu aloitusikkunassa olevaan painikkeeseen, jolla siirrytään valikosta eteenpäin (kuva 14).

```
void Start ()
{
    inputField = GetComponent<TMP_InputField>();

    if (inputField != null)
    {
        if (PlayerPrefs.HasKey(playerNamePrefKey))
        {
            defaultName = PlayerPrefs.GetString(playerNamePrefKey);
            inputField.text = defaultName;
        }
    }

    PhotonNetwork.playerName = defaultName;
}


public void SetPlayerName(string name)
{
    PhotonNetwork.playerName = name + " ";
    PlayerPrefs.SetString(playerNamePrefKey, name);
}
```

Kuva 14. Nimen syöttäminen.

Hahmovalinnassa (kuva 16) käytettiin hyväksi listaa, johon kaikki hahmot on asetettu. Hahmot ovat asetettu ”CharacterContainer”-pelibojektiin, joka sisältää jokaisesta hahmosta graafisen kuvakkeen. Jokaiselle hahmolle on oma painike, joka syöttää arvon ja muuttaa hahmon tyyppiä etukäteen (kuva 15). Kun peli alkaa, hahmonluontitoiminnallisuus tietää mitä hahmoa pelaaja haluaa käyttää ja luo pelaajan peliin.

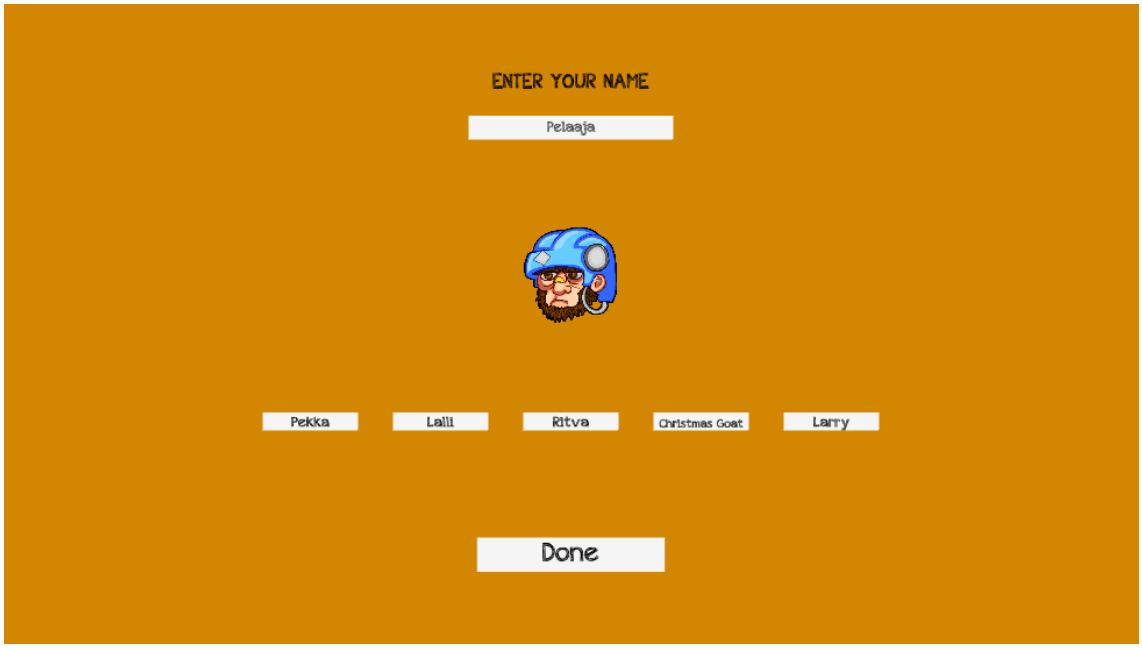
```
public void SelectCharacter(int index)
{
    if(index == selectionIndex) return;

    if (index < 0 || index >= characterList.Count) return;

    characterList[selectionIndex].SetActive(false);
    selectionIndex = index;
    characterList[selectionIndex].SetActive(true);

    MenuOptions.Instance.CharacterType = (CharacterType) selectionIndex;
}
```

Kuva 15. Hahmovalintaan tehty toiminnollisuus.



Kuva 16. Hahmonvalintaikkuna.

5.3.3 Huoneiden luonti, listaus ja peliin liittyminen

Huoneiden luonti tapahtuu aulasta. Aulassa on erillinen painike nimeltä “Create room”. Painike hävittää huonelistauksen ja tuo esille valikon, mistä voidaan asettaa huoneen nimi, huoneen yksityisyys sekä maksimipelaajamäärä. Valikosta voidaan palata taaksepäin painamalla “Cancel”-painiketta ja luoda huone “Create room”-painikkeella. Kyseinen painike vie pelaajan suoraan luotuun huoneeseen.

Huoneen nimi, yksityisyys sekä maksimipelaajamäärä asetetaan huoneen asetuksiin niistä valituista kentistä. Huoneen luonti toteutetaan, mikäli käyttäjä on yhdistänyt Photonin verkkoon ja huoneen nimi-tekstikenttä ei ole tyhjänä. Huoneen luonnissa käytetään *PhotonNetwork.CreateRoom*-funktiota (kuva 17), joka luo huoneen verkkoon. Huoneelle voidaan antaa myös muokattuja tietoja, joita seurataan pelin käynnissä ollessa. Kyseessä voi olla esimerkiksi sisällytettynä pisteseuranta, joka laskee kunkin pelaajan saamia pisteitä. Opinnäytetyön kehitysvaiheessa ominaisuutta ei kuitenkaan vielä otettu käyttöön, mutta se otettiin huomioon suunnitellessa verkkopeliä. (First Gear Games 2017.)

```
public void OnClickCreateRoom()
{
    RoomOptions roomOptions = new RoomOptions() {IsVisible = isRoomVisible, IsOpen = true, MaxPlayers = maxPlayerAmount};
    print(isRoomVisible + " " + maxPlayerAmount);
    if (PhotonNetwork.connected && roomName.text != string.Empty)
    {
        PhotonNetwork.CreateRoom(roomName.text, roomOptions, TypedLobby.Default);
        Debug.Log("Room " + roomName.text + " Successfully Created");
    }
    else
    {
        Debug.Log("Room not created");
    }
}
```

Kuva 17. Huoneen luonti.

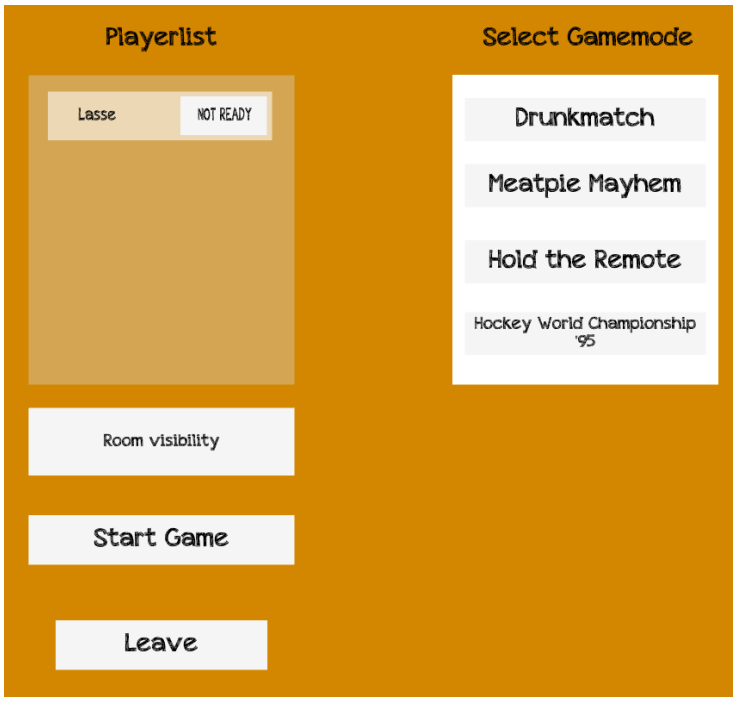
Huoneet listataan aulassa isoon näkymään ja vie suurimman osan aulan ulkoasusta. Listassa näkyvät vain ne huoneet, jotka ovat avoimia ja/tai ne eivät ole täynnä (kuva 18). Jokainen listattu huone on painikekomponentti ja niihin voidaan liittyä painamalla huoneen nimeä. Jokaisen huoneen kohdalla listataan huoneen nimi, missä palvelimella alueellisesti se sijaitsee sekä pelaajamäärä. Päätimme asettaa alueen listaukseen, jotta pelaajat tietävät onko yhteys kyseisen huoneeseen hyvä vai huono. Vaihtoehtona olisi myös lisätä näkymään ”ping”, eli viive palvelimen ja pelaajan välillä, mutta emme nähneet sitä tarpeelliseksi tehdä vielä tässä vaiheessa. Tärkeintä on tietää, mikä on huoneen maksimipelaajamäärä ja kuinka monta pelaajaa siellä tällä hetkellä on. Huone on vain näkyvillä, mikäli käyttäjä on asettanut sen julkiseksi tai huoneessa on vielä tilaa.

Kuva 18. Huonenäkymä.

Mikäli käyttäjä päättää luoda yksityisen huoneen (Friends only), huonetta ei listata aulan näkymässä. Huoneen luoja voi kuitenkin myöhemmin vaihtaa huoneen näkyvyyden julkiseksi. Mikäli luotu huone ei ole julkinen, voivat muut pelaajat liittyä huoneeseen “Join private game”-painikkeella. Painike avaa uuden ikkunan, jossa on tekstikenttä salaisen huoneen nimeä varten. Muiden pelaajien tulee pyytää huoneen luojalta nimi liittyäkseen peliin.

5.3.4 Huoneen sisältö

Huone on ns. toinen odotustila mihin pelaajat tulevat ennen pelin alkamista (kuva 19). Huone sisältää listauksen pelaajista, pelin käynnistämiseen tarvittavan painikkeen, kenttävalinnan, poistumispainikkeen sekä painikkeen, jolla voidaan muuttaa, onko huone julkinen vai salainen.



Kuva 19. Huoneen näkymä.

Osa painikkeista on käytössä vain huoneen luojalle. Huoneen luoja, eli master-käyttäjä voi ainoastaan käynnistää pelin, vaihtaa pelikentän, muuttaa pelin näkyvyyden (julkinen/suljettu) sekä potkia pois huoneesta muita pelaajia klikkaamalla heidän nimeään. Jokaisella pelaajalla on myös oma valmiustilan ilmaiseva painike, jota klikkaamalla pelaajat voivat ilmoittaa olevansa valmiita aloittaakseen pelin.

”Ready”-painikkeen toteutus oli oma haasteensa. Aluksi pelaajien luonti toteutettiin paikallisesti, hakemalla palvelimelta pelaajat uudestaan listaan pelaajan liittyessä huoneeseen. Mikäli pelaaja poistui, poistettiin peliobjekti jokaiselta paikallisesti. Toteutustapa ei sopinut ”ready”-painikkeen kanssa, sillä jokaisen pelaajan tulisi hallita omaa painiketta ja tässä toteutustavassa kaikki painikkeet kuuluivat aina vain paikalliselle pelaajalle.

Paras vaihtoehto oli luoda verkon yli painikkeen peliobjekti ja luoda aina uusi Photon View -komponentti kyseiselle objektille. Uuden Photon View -komponentin luomisen yhteydessä siirrettiin sen omistajuus liittyneelle pelaajalle *PhotonView.TransferOwnership* komennolla. Koska peliobjektit luotiin verkon yli, erillistä peliobjektin poistamista ei tarvittu. Tämä johtuu siitä, kun pelaaja poistuu huoneesta, kaikki sen omistamat peliobjektit poistetaan automaattisesti.

5.4 Pelin toiminnallisuudet ja tekniikat

Kaikki pelaajat ovat edelleen samassa huoneessa pelin käynnistyttyä. Pelaajien kaikkea liikkumista ja käyttäytymistä mallintavat toiminnallisuudet välitetään verkon yli muille pelaajille. Tiedon välittämiseen voidaan käyttää kolmea eri toteutustapaa: remote procedure calls (RPC), OnPhotonSerializeView, eli jatkuva tiedon lähettäminen sekä RaisedEvents-tapahtumakutsut.

5.4.1 Remote Procedure Calls

RPC-kutsut mahdollistavat funktioiden kutsumisen ja lähettämisen verkon ylitse, kun niitä kutsutaan. Mikäli RPC-kutsuja käytetään, tulee käytettävän funktion eteen asettaa [PunRPC]-attribuutti (kuva 20). Tämä lisää funktion palvelintiedoston (PhotonNetworkSettings) tietoihin ja palvelin osaa käsitellä saadut pyynnöt oikein. Kun tätä funktiota halutaan kutsua ja lähettää tietoa verkon ylitse, kutsutaan apufunktiota mikä suorittaa funktion toiminnallisuuden. RPC-funktion lähetys tapahtuu kutsumalla sitä seuraavanlaisesti (Exit Games 2018j.):

photonView.RPC("<funktion nimi>", <kenelle se lähetetään>, funktion parametri1, parametri2, parametri x);

RPC-kutsuja on hyvä hyödyntää, kun halutaan lähettää tietoa verkon yli vain tietyn muutoksen tapahtuessa ja kun kyseinen muutos tapahtuu vain harvakseltaan. RPC:n avulla voidaan myös lähettää tietoa vain tietyille pelissä oleville pelaajille. Esimerkiksi tieto vahingon ottamisesta voidaan välittää juuri oikealle pelaajalle. Tällaisessa

tilanteessa välitetään tieto, kuka vahingon tekijä on ja paljon osuma tekee vahinkoa. Tällöin projektiililla tulee olla Photon View-komponentti ja sille määrätty ”omistaja”. RPC-kutsut voidaan myös tallentaa (bufferoida) muistiin, jolloin jokainen myöhässä tullut pelaaja saa tiedon tapahtuneesta.

RPC-kutsuilla toteutetaan lähestulkoon kaikki muu paitsi pelaajan sijainnin seuranta. RPC-kutsuilla luodaan kenttään kerättävät aselaatikot, ammukset sekä kaikki muut peliobjektit, joita ei tarvitse seurata koko ajan. Ohessa esimerkkitoteutus hyppy-animaation lähettämisestä RPC-toiminnon avulla (kuva 20).

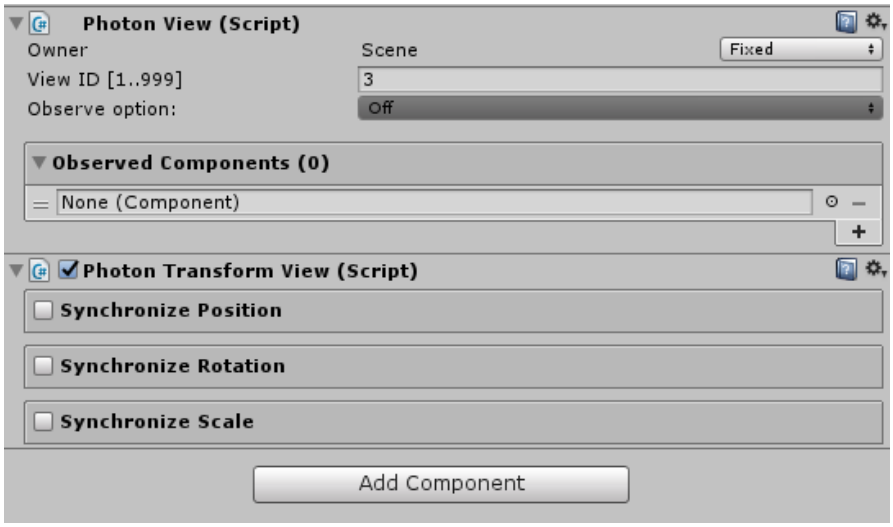
```
void UpdateJumping()
{
    if (Input.GetButton("Jump") && m_IsGrounded)
    {
        m_Animator.SetTrigger("IsJumping");
        m_Body.AddForce(Vector2.up * JumpForce);
        m_PhotonView.RPC("DoJump", PhotonTargets.Others);
    }
}

[PunRPC]
void DoJump()
{
    m_Animator.SetTrigger( "IsJumping" );
}
```

Kuva 20. RPC-funktio hyppyanimaation lähettämiselle. Firetail Games 2018.

5.4.2 Jatkuva tiedonlähetys

Pelaajien sijainnista tulee jatkuvasti lähettää tietoa muille pelaajille, joten RPC-kutsua ei ole järkevä kyseisessä tilanteessa käyttää. PUN-työkalulla voidaan helposti seurata pelaajien liikkeitä lisäämällä Photon Transform View-komponentti (PTV) peliobjektiin (kuva 21), jonka liikettä halutaan seurata. Komponentti vaatii myös Photon View-komponentin (PV) ja PTV-komponentti pitää lisätä sen seurattavaksi. (Exit Games 2018k.)



Kuva 21. Photon View -komponentti.

PV-komponentti pitää olla jokaisella peliobjektilla, jonka tietoa halutaan viedä verkon ylitse. PV-komponentti yksilöi jokaisen pelaajan ja peliobjektin mitä kentällä nähdään ja mikä vie tietoa verkon ylitse. PV-komponentti sisällyttää tiedon sen omistajasta, sen ID-arvosta sekä mitä komponentteja se seuraa.

Photon View-komponentissa on mahdollista seurata peliobjektien muutosta ja ne voidaan ja tulee tietyissä tilanteissa lisätä ”Observed Components”-listaan. PV-komponentista voidaan muuttaa tapaa, milloin se päivittää tapahtuvan muutoksen verkon yli. Päivitystapa voidaan vaihtaa ”Observe option”-pudotusvalikosta. (Exit Games 2018l.)

Mikäli valikosta valitaan ”Off”-tila, komponentin seuraaminen on pois päältä. Tätä tilaa on hyvä käyttää, jos tietoa lähetetään vain RPC-kutsuja käyttämällä. (Exit Games 2018l.)

”Unreliable” tilassa lähetetään tieto ”sellaisenaan”. Päivitykset lähetetään heti kun ne tapahtuvat. Unrelable-tila on hyvä valita, kun lähetetään esimerkiksi pelaajan sijaintia tai absoluuttista dataa. Se ei kuitenkaan sovellu triggereille kuten aseiden vaihtamiselle. Kyseistä tilaa pitää käyttää mahdollisimman vähän, sillä se lähettää tietoa peliobjektin sijainnista, vaikka peliobjekti ei liikkuisikaan. (Exit Games 2018l.)

”Unreliable on Change” tarkistaa jokaisen päivityksen, tapahtuuko siinä muutoksia. Jos kaikki tieto pysyy samana, lähetetään vain yksi päivitys, että tila ei ole muuttunut ja datan lähettäminen lopetetaan. Tätä on hyvä käyttää, kun peliobjekti voi pysähtyä tai halutaan

toistaa vain siinä tilanteessa, kun jotain muutoksia kyseiselle peliobjektille tapahtuu. (Exit Games 2018l.)

”Reliable Delta Compressed” tila vertailee jokaisen lähetetyn päivityksen arvoa edelliseen. Mikäli arvot eivät ole muuttunut edellisestä päivityksestä uusi päivitys ohitetaan. (Exit Games 2018l.)

Photon View-komponentin omistaja näkyy komponentin ylälaidassa. Mikäli komponentti allokoidaan uudelle käyttäjälle, omistajuus asetetaan automaattisesti yhdistäneelle pelaajalle. Omistajuutta voidaan myös vaihtaa tai asettaa lennosta.

Omistajuuden vaihtaminen toteutetaan koodin puolella ja silloin PV-komponentin omistajuustila täytyy olla joko ”Takeover” tai ”Request”. ”Fixed”-tilassa olevan PV-komponentin omistajaa ei voida vaihtaa kuin kerran sitä allokoidessa. Tämän tilan voi vaihtaa PV-komponentin oikeasta yläkulmasta löytyvästä pudotusvalikosta.

Mikäli halutaan seurata jatkuvasti jonkin komponentin muuttumista, tulee käyttää IPunObservable-rajapintaa (kuva 22). Kyseinen rajapinta seuraa koko ajan mitä tietoa lähetetään ja vastaanotetaan. Tieto on kuitenkin rajattua ja pelikehittäjän tulee itse määrittää mitä tietoa liikutetaan. Kun rajapinta on otettu käyttöön, tulee vielä sen pakollinen metodi *OnPhotonSerializeView* ottaa käyttöön (kuva 23). Tämän metodin alle lisätään kaikki mitä tietoa halutaan lähettää ja vastaanottaa. (Exit Games 2018k.)

```
public class TrackPlayer : MonoBehaviour, IPunObservable
{
```

Kuva 22. IPunObservable rajapinnan lisääminen.

```
public void OnPhotonSerializeView(PhotonStream stream, PhotonMessageInfo info)
{
    if (stream.isWriting)
    {
        stream.SendNext(transform.position);
        stream.SendNext((byte)this._characterState);
    }
```

Kuva 23. Seurattavat ja lähetettävät komponentit.

Kaikki mitä kyseisen metodin avulla seurataan tai lähetetään, lähetetään useasti sekunnissa. Oletuksena PUN lähettää tietoa palvelimelle 25 kertaa sekunnissa. Mikäli tiedon lähettämisen tiheyttä halutaan kasvattaa tai vähentää, voidaan sen arvoa muokata PUN-työkalun asetuksista tai sen arvoa voidaan muuttaa Photon-verkkoon yhdistävässä skriptassa (kuva 24). (Exit Games 2018k.)

```
private const int sendrate = 50;
private bool isInLobby;
private void Awake()
{
    PhotonNetwork.sendRate = sendrate;
    PhotonNetwork.sendRateOnSerialize = sendrate;
}
```

Kuva 24. Lähetyksen tiheyden asettaminen.

5.4.3 RaiseEvent-tapahtumakutsut

RPC-kutsut ja jatkuvasti seurattavat muutokset eivät kuitenkaan välttämättä ole aina paras mahdollinen ratkaisu. PUN-työkalun sisältämä RaiseEvent-toiminnollisuus (RE) ei vaadi PhotonView-komponenttia ja voidaan käyttää samaan tapaan kuin mitä tahansa muutakin tapahtumakutsua. (Exit Games 2018j.)

Tapahtuman sisältö voi olla mitä tahansa tietoa, mitä PUN pystyy käsittelemään. Kuten normaalienkin tapahtumakutsujen tapaan, RE-toiminnolle tulee luoda takaisinkutsu-funktio, joka seuraa tapahtumien muutosta. RE-toiminnon voi myös RPC-kutsun tapaan määrittää kenelle tapahtuman muutokset lähetetään. Lähetetäänkö tieto muille (kuin itselleen), kaikille vai pelkästään pelaajalle, joka on vastuussa pelissä tapahtuvista muutoksista. Tapahtumamuutokset voidaan myös lisätä huoneen välimuistiin. Tämä mahdollistaa sen, että myöhemmin saapuneet pelaajat saavat tapahtuneista tiedon liityttyään peliin. RE-tapahtumakutsua ei kuitenkaan tämän opinnäytetyön aikana käytetty kertaakaan. (Exit Games 2018j.)

6 Tulokset

Opinnäytetyön valmistumisvaiheessa läheskään kaikkia pelissä käytettäviä ominaisuuksia ei saatu valmiiksi. Jatkossa peliä kehittävät työntekijät ovat kuitenkin tietoisia, kuinka PUN-työkalun ominaisuuksia tulee käyttää tietyissä tilanteissa.

Opinnäytetyön aikana tuli useita haasteita vastaan. Suurin haaste oli tutustua uuteen työkaluun ja sen toiminnollisuuksien haltuun ottaminen. Isona haasteena oli saada pelaajat kommunikoimaan pelissä toisilleen, koska kommunikointiin voidaan käyttää eri tapoja, ja aina ei välttämättä ole täysin selvää mitä tapaa tietyissä tilanteissa olisi paras käyttää.

Ongelmaksi nousi myös huoneen odotustilassa kommunikointi ”Ready”-painikkeella. Koska Photon-verkkoon ja aulaan liittyessä pelaajalle ei luoda objektia, jota pelaaja ohjaa, tuli pelaajan yhdistäessä huoneeseen luoda peliobjekti, jonka pelaaja omistaa ja vain sen omistaja voisi reagoida painikkeella.

Opinnäytetyön aikana saatiin valmiiksi toimiva alusta verkkomoninpelille. Tärkein valmistunut kokonaisuus opinnäytetyössä oli verkkomoninpeliin pääsemisen prosessi. Pelaaja voi nyt liittyä suoraan verkkopeliin valikosta ja luoda tai liittyä olemassa olevaan verkkopeliin. Pelaajat pääsevät samaan peliin ja pystyvät liikkumaan ja näkemään toisensa pelissä. Prosessissa päästiin myös haluttuun yksinkertaiseen ja nopeaan tapaan päästä pelaamaan. Pelaajien on yksinkertaista luoda huoneita ja liittyä peliin. Valikoiden välillä edestakaisin toimiminen on nopeaa ja helppoa.

Jatkokehitysmahdollisuuksia peliin jäi runsaasti. Ulkoasu jäi todella pelkistetyksi, aulassa huoneiden listauksessa voisi alueen sijaan olla viiveen suuruus huoneeseen ja huoneessa peli voisi alkaa, kun kaikki pelaajat ovat painaneet ”Ready”-painiketta.

Opinnäytetyön tavoitteet täyttyivät opinnäytetyön omalta osaltani. PUN-työkalu tuli tutuksi, Unity3D-osaaminen kehittyi runsaasti sekä verkkomoninpeli saatiin hyvälle alulle jatkoa varten. Olen myös tyytyväinen, että näinkin nopealla aikataululla sain toteutettua toimivan käyttöliittymän verkkopelaamista varten. Olisin kuitenkin opinnäytetyön aikana

halunnut saada vielä toimivamman kokonaisuuden, mutta aikataulun vuoksi se ei ikävä kyllä ollut mahdollista.

7 Pohdinta

PUN ja verkkomoninpelin toteutus aiheena osoittautui mielenkiintoisaksi ja tulen jatkossa omissa projekteissa kehittämään omaa osaamistani työkalulla. Oli myös loistavaa päästä toteuttamaan toiminnollisuutta jo valmiiseen projektiin, koska työn tulokset olivat nähtävissä heti. Kokonaisuutena opinnäytetyön aihe kehitti paljon osaamista verkkomoninpelien toteutuksesta.

PUN ja muut Photon Enginen tarjoamat tuotteet ovat varteenotettavia vaihtoehtoja verkkomoninpelin kehitykseen. Tuotteet antavat kehittäjälle paljon vapauksia toteuttaa toiminnollisuuksia ja ainakin PUN-työkalun avulla on helppo lähteä toteuttamaan verkkomoninpeliä. Tuotteet kilpailevat myös hyvin muiden samanlaisten palveluntarjoajien kanssa ja he tarjoavat monipuolisia tuotteita.

PUN-työkalu on helppo asentaa ja aloittaa moninpelin toteutus. Kokonaisuuden toteuttaminen kuitenkin vaatii paljon tekemistä ja huolellisuutta, jotta toteutuksesta saataisiin mahdollisimman onnistunut. Aluksi työkalun käyttäminen tuntui hankalalta ymmärtää ja vaati useiden eri dokumenttien ja esimerkkien läpikäymistä. Laaja ja hyvin toteutettu dokumentointi auttoi kuitenkin työkalun haltuunotossa ja alkuun pääsi suhteellisen vaivattomasti.

PUN sisältää paljon valmiita toiminnollisuuksia, joita kannattaa hyödyntää. Kannattaa kuitenkin olla aina tietoinen, mitä mikäkin toiminto tekee taustallaan. Esimerkiksi Photon-verkon kautta peliobjektien instantiointi ei välttämättä aina ole järkevin ratkaisu, vaan instantiointi kannattaa tehdä jokaisella pelaajalla paikallisesti.

PUN toimii myös erinomaisesti yhteen Unity3D-pelimoottorin kanssa. Asennus on yksinkertaista ja opinnäytetyön aikana ei mitään ristiriitoja näiden kahden välillä

huomattu. PUN sisältää myös valmiita Unity-komponentteja, joiden avulla on helppoa muokata ja asettaa tiettyjä toiminnolisuuksia.

Lähteet

Amazon. 2018. AWS for Gaming. https://aws.amazon.com/gaming/. 25.3.2018.

Exit Games. 2018a. Photon Engine. https://www.photonengine.com/en-US/Photon. 25.3.2018.

Exit Games. 2018b. Photon Powered Games. https://www.photonengine.com/en/realtime/showcase. 25.3.2018.

Exit Games. 2018c. Photon Cloud or Photon Server? https://doc.photonengine.com/en-us/realtime/current/getting-started/onpremises-or-saas. 25.3.2018.

Exit Games. 2018d. Photon Bolt. https://www.photonengine.com/en/BOLT. 25.3.2018.

Exit Games. 2018e. Photon PUN. https://www.photonengine.com/en/PUN. 25.3.2018.

Exit Games. 2018f. Photon Quantum. https://www.photonengine.com/en/quantum. 25.3.2018.

Exit Games. 2018g. Pricing. https://www.photonengine.com/en-US/PUN/pricing#plan-20. 25.3.2018.

Exit Games. 2018h. Photon Engine Regions. https://doc.photonengine.com/en-us/pun/current/connection-and-authentication/regions. 19.5.2018.

Exit Games. 2018i. Initial Setup. https://doc.photonengine.com/en-us/pun/current/getting-started/initial-setup. 21.5.2018.

Exit Games. 2018j. RPC and RaiseEvent. https://doc.photonengine.com/en-us/pun/current/gameplay/rpcsandraiseevent. 21.5.2018.

Exit Games. 2018k. PUN basics tutorial – Player Networking https://doc.photonengine.com/en-us/pun/current/demos-and-tutorials/pun-basics-tutorial/player-networking. 21.5.2018.

Exit Games. 2018l. Feature Overview. https://doc.photonengine.com/en-us/pun/current/getting-started/feature-overview. 23.5.2018.

David Xicota, Gamedonia. 2018. Lag compensation techniques for multiplayer games in realtime. http://www.gamedonia.com/blog/lag-compensation-techniques-for-multiplayer-games-in-realtime. 19.5.2018.

Robert, Exit Games. 2017. Photon Quantum – Join the Multiplayer Revolution. https://blog.photonengine.com/2017/12/21/photon-quantum/. 25.3.2018

Fernando Bevilacqua, Tutsplus. 2013. Building a Peer-toPeer Multiplayer Networked Game. https://gamedevelopment.tutsplus.com/tutorials/building-a-peer-to-peer-multiplayer-networked-game--gamedev-10074. 19.5.2018.

Unity. 2018. Unity Multiplayer. https://unity3d.com/unity/features/multiplayer. 25.3.2018.

Wegmann, C. 2017. Photon vs UNET: Battle of the Giants. https://www.slideshare.net/ChristofWegmann/photon-vs-unet-battle-of-the-giants. 25.3.2018.

Wikipedia. 2016. Exit Games. https://en.wikipedia.org/wiki/Exit_Games. 25.3.2018.

Wikipedia. 2018. Multiplayer Video Game. https://en.wikipedia.org/wiki/Multiplayer_video_game. 19.5.2018.

XtraLife. 2018. XtraLife Features. http://xtralife.cloud/features/. 25.3.2018.

First Gear Games. Youtube. 2017. Photon Networking. https://www.youtube.com/watch?v=EqW-4r7goaQ&list=PLkx8oFug638qVMIrtqOnwmqnW6o8WDgQ1. 21.5.2018.